Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра системотехніки

Дисципліна: «СПЕЦІАЛІЗОВАНІ БАЗИ ДАНИХ (NOSQL)»

**ПРАКТИЧНЕ ЗАНЯТТЯ № 1-2**

**«ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМИ РОБОТИ З БД»**

**«ПРОЕКТУВАННЯ І РОЗРОБКА ШАРУ DAO ДЛЯ РОБОТИ З БД»**

|  |  |  |
| --- | --- | --- |
| Виконав:  ст. гр. ІТКН-18-2 |  | Прийняв:  Доц. каф. Хряпкін О.В. |

Харків 2020

1. **Мета роботи**

Спроектувати і розробити базу даних з відповідною структурою (таблиці, зв'язку) відповідно до обраної предметної області. Отримати практичні навички: аналізу предметної області та проектування на цій основі структури БД. Отримати практичні навички: аналізу предметної області та проектування на цій основі структури БД, створення на основі шаблону проектування DAO модуля взаємодії програми та обраної СУБД.

1. **Постановка задачі**

Спроектувати і реалізувати програму по зберіганню і управлінню даними комп'ютерних принтерів.

1. Розробити Use Case діаграму і детальний опис кожного варіанту використання (яка інформація необхідна для виконання Use Case, що є результатом). Оформити у вигляді таблиці отримані Use Case. Кількість варіантів використання не менше 5.

2. Провести аналіз предметної області. Спроектувати і створити структуру БД (таблиці і зв'язку). Результат - готова БД і її структура у вигляді діаграми таблиць і зв'язків. Мінімальна кількість таблиць 3.

3. Для обраної предметної області написати модуль DAO. Для цього створити в проекті окремий пакет «DAO» і помістити в нього основні файли модуля: інтерфейс IMyDAO, клас DAOFactory, перерахування «enum», яке містить існуючі реалізовані види DAO і клас реалізації інтерфейсу «IMyDAO» MySQLDAO з усіма методами. Приклад реалізації пакету DAO знаходиться в проекті ExampleDAO.zip.

4. Проаналізувати методи DAO, визначити ті при виклику яких можуть виникати помилки (колізії при паралельному виконанні запитів до БД) і вибрати для них відповідний рівень ізоляції транзакції.

5. Перевірити працездатність розробленого модуля DAO.

6. Оформити звіт з виконання завдання.

**3. Опис предметної області**

В якості предметної області виступає система по зберіганню і управлінню даними комп'ютерних моніторів. Таблиця Monitor повинна містити наступну інформацію : id, Name, Matrix\_type, Aspect\_ratio, Diagonal\_matrix, Screen\_resolution, Screen\_refresh\_rate, Type(Object), Model(Object). Таблиця Type повинна містити наступну інформацію : id, Name\_type, Features. Таблиця Model повинна містити наступну інформацію : id, Name\_company, Country, Name\_model.

**4. Хід роботи**

**4.1 Use Case діаграма**

В ході аналізу предметної області була побудована Use Case діаграма що зображена на рисунку 1.

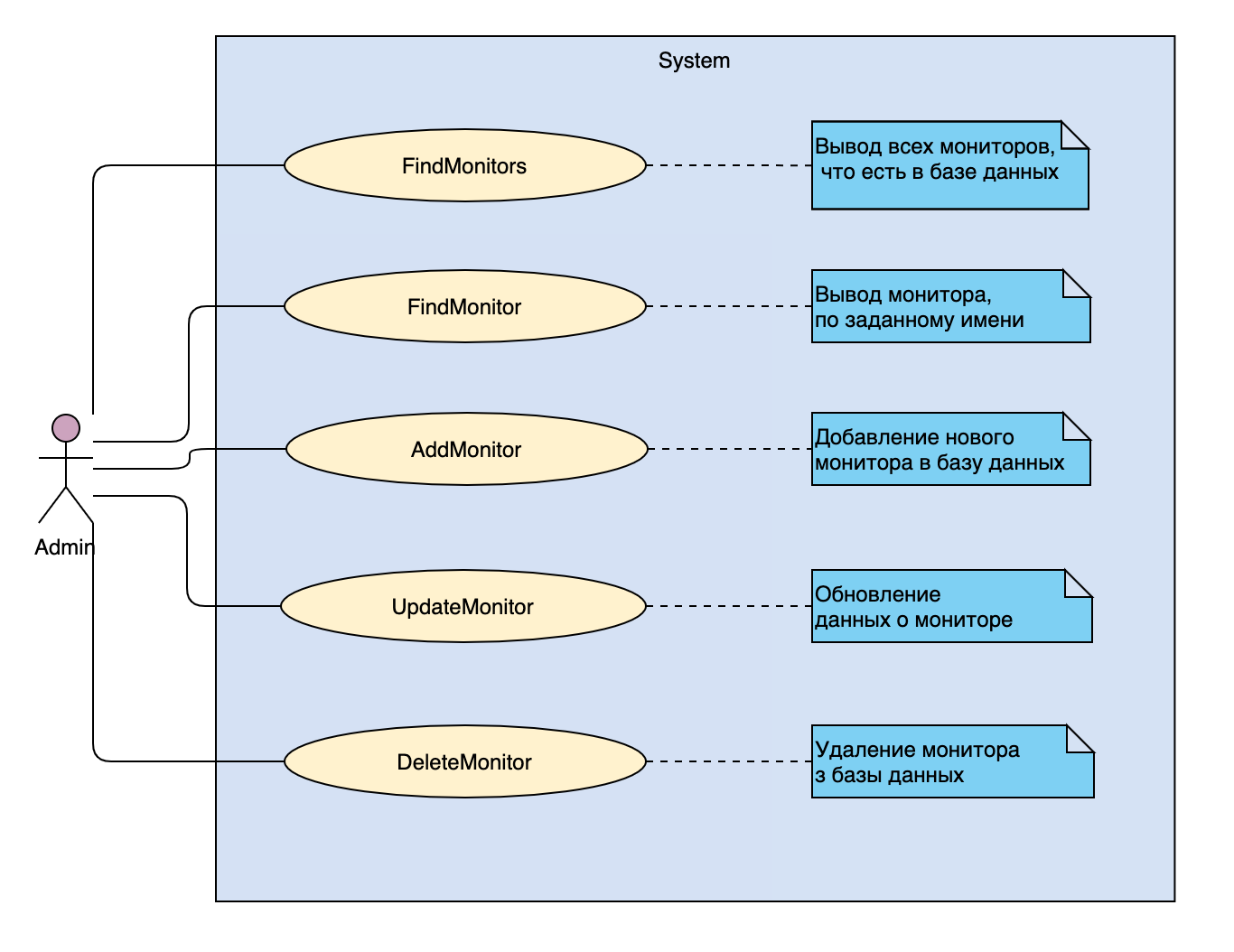


Рисунок 1 – Use case діаграма

**4.2 ER-діаграма**

В наслідок аналізу предметної області була побудована ER-діаграма, що представлена на рисунку 2.

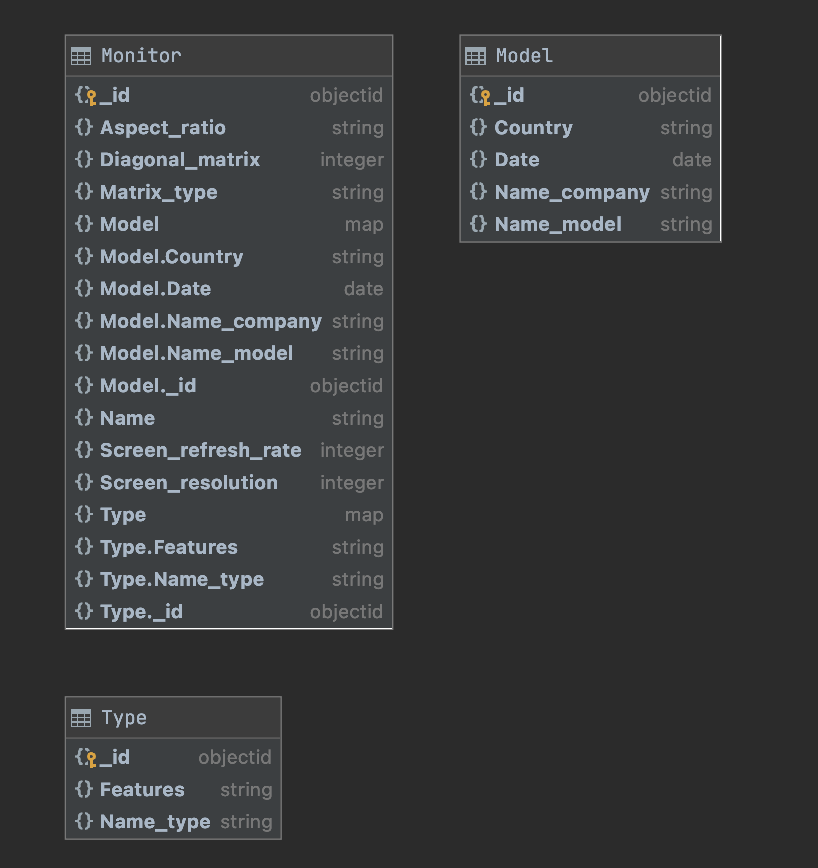


Рисунок 2 – ER-діаграма бази даних

**4.3 Перевірка працездатності розробленого модуля DAO:**

**DaoFactory**

package Dao;  
  
import Impl.ModelDaoImpl;  
import Impl.MonitorDaoImpl;  
import Impl.TypeDaoImpl;  
  
public class DaoFactory implements IDaoFactory{  
  
 @Override  
 public MonitorDao GetMonitorDao() {  
 return new MonitorDaoImpl();  
 }  
  
 @Override  
 public ModelDao GetModelDao() {  
 return new ModelDaoImpl();  
 }  
  
 @Override  
 public TypeDao GetTypeDao() {  
 return new TypeDaoImpl();  
 }  
}

**IDaoFactory**

package Dao;  
  
interface IDaoFactory {  
  
 MonitorDao GetMonitorDao();  
 ModelDao GetModelDao();  
 TypeDao GetTypeDao();  
}

**MonitorDao**

package Dao;  
  
import Models.Monitor;  
import org.bson.types.ObjectId;  
  
import java.util.List;  
  
public interface MonitorDao {  
  
 void getAllMonitors();  
 void getByName(String Name);  
 void createMonitor(Monitor monitor);  
 void updateMonitor(Monitor monitor);  
 void deleteMonitor(Monitor monitor);  
  
}

**MonitorDaoImpl**

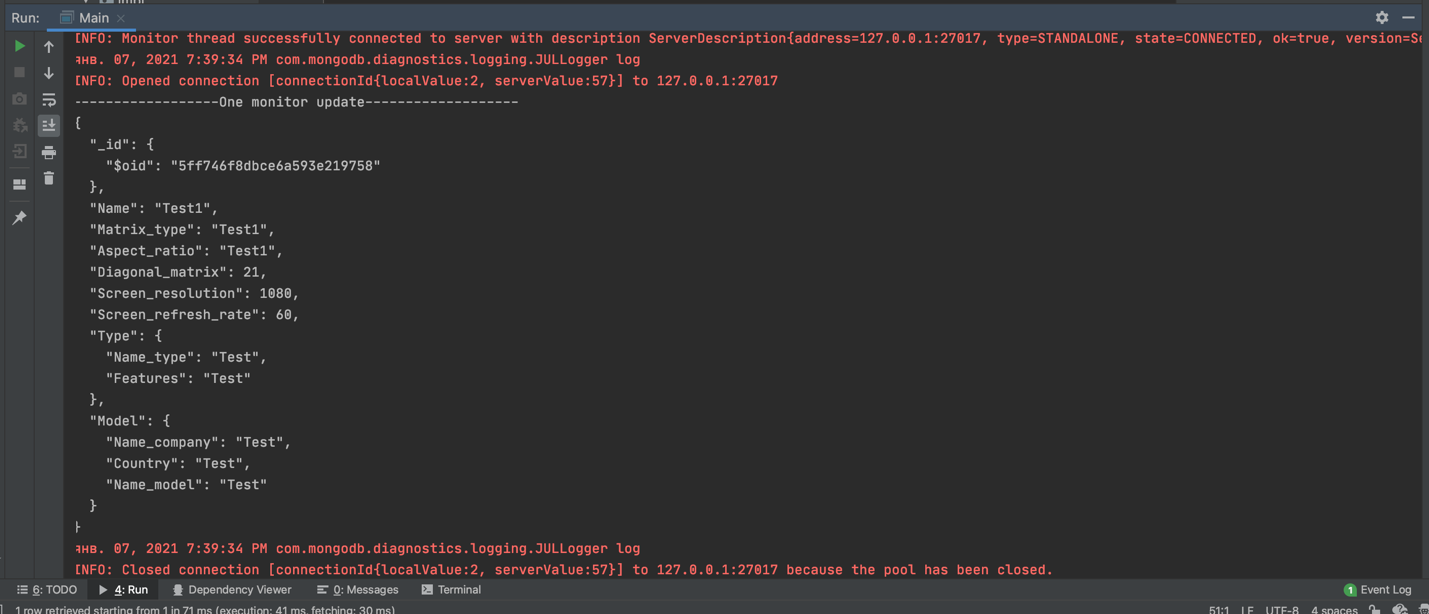
package Impl;  
  
import Dao.MonitorDao;  
import Models.Monitor;  
import com.mongodb.client.MongoClient;  
import com.mongodb.client.MongoClients;  
import com.mongodb.client.MongoCollection;  
import com.mongodb.client.MongoDatabase;  
import com.mongodb.client.result.DeleteResult;  
import org.bson.Document;  
import org.bson.conversions.Bson;  
import org.bson.json.JsonWriterSettings;  
import org.bson.types.ObjectId;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import static com.mongodb.client.model.Filters.*eq*;  
  
public class MonitorDaoImpl implements MonitorDao {  
  
 JsonWriterSettings prettyPrint = JsonWriterSettings.*builder*().indent(true).build();  
  
 @Override  
 public void getAllMonitors() {  
 try (MongoClient mongoClient = MongoClients.*create*("mongodb://127.0.0.1:27017")) {  
 MongoDatabase DB = mongoClient.getDatabase("Monitors");  
 MongoCollection<Document> Collection = DB.getCollection("Monitor");  
  
 List<Document> monitorList = Collection.find().into(new ArrayList<>());  
 System.*out*.println("Monitors: ");  
 for (Document monitor : monitorList) {  
 System.*out*.println(monitor.toJson());  
 }  
 System.*out*.println("------------------Monitors found-------------------");  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void getByName(String Name) {  
 try (MongoClient mongoClient = MongoClients.*create*("mongodb://127.0.0.1:27017")) {  
 MongoDatabase DB = mongoClient.getDatabase("Monitors");  
 MongoCollection<Document> Collection = DB.getCollection("Monitor");  
  
 Document Monitor\_getByName = Collection.find(new Document("Name", Name)).first();  
 System.*out*.println("Monitor: " + Monitor\_getByName.toJson());  
 System.*out*.println("------------------One monitor found-------------------");  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void createMonitor(Monitor monitor) {  
 try (MongoClient mongoClient = MongoClients.*create*("mongodb://127.0.0.1:27017")) {  
 MongoDatabase DB = mongoClient.getDatabase("Monitors");  
 MongoCollection<Document> Collection = DB.getCollection("Monitor");  
  
 Document Monitor\_createMonitor = new Document("\_id", new ObjectId());  
 Document Monitor\_createTypeMonitor = new Document("\_id", new ObjectId());  
 Document Monitor\_createModelMonitor = new Document("\_id", new ObjectId());  
  
 Monitor\_createTypeMonitor.append("Name\_type", monitor.Type.getName\_type())  
 .append("Features", monitor.Type.getFeatures());  
  
 Monitor\_createModelMonitor.append("Name\_company", monitor.Model.getName\_company())  
 .append("Country", monitor.Model.getCountry())  
 .append("Name\_model", monitor.Model.getName\_model());  
  
 Monitor\_createMonitor.append("Name", monitor.getName())  
 .append("Matrix\_type", monitor.getMatrix\_type())  
 .append("Aspect\_ratio", monitor.getAspect\_ratio())  
 .append("Diagonal\_matrix", monitor.getDiagonal\_matrix())  
 .append("Screen\_resolution", monitor.getScreen\_resolution())  
 .append("Screen\_refresh\_rate", monitor.getScreen\_refresh\_rate())  
 .append("Type", Monitor\_createTypeMonitor)  
 .append("Model", Monitor\_createModelMonitor);  
  
 Collection.insertOne(Monitor\_createMonitor);  
 System.*out*.println("------------------One monitor inserted-------------------");  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void updateMonitor(Monitor monitor) {  
 try (MongoClient mongoClient = MongoClients.*create*("mongodb://127.0.0.1:27017")) {  
 MongoDatabase DB = mongoClient.getDatabase("Monitors");  
 MongoCollection<Document> Collection = DB.getCollection("Monitor");  
  
 Document setData\_Type = new Document();  
 setData\_Type.append("Name\_type", monitor.Type.getName\_type())  
 .append("Features", monitor.Type.getFeatures());  
  
 Document setData\_Model = new Document();  
 setData\_Model.append("Name\_company", monitor.Model.getName\_company())  
 .append("Country", monitor.Model.getCountry())  
 .append("Name\_model", monitor.Model.getName\_model());  
  
 Bson filter = *eq*("\_id", monitor.getId());  
 Document setData\_Monitor = new Document();  
 setData\_Monitor.append("Name", monitor.getName())  
 .append("Matrix\_type", monitor.getMatrix\_type())  
 .append("Aspect\_ratio", monitor.getAspect\_ratio())  
 .append("Diagonal\_matrix", monitor.getDiagonal\_matrix())  
 .append("Screen\_resolution", monitor.getScreen\_resolution())  
 .append("Screen\_refresh\_rate", monitor.getScreen\_refresh\_rate())  
 .append("Type", setData\_Type)  
 .append("Model", setData\_Model);  
  
 Document update = new Document();  
 update.append("$set", setData\_Monitor);  
  
 Collection.updateOne(filter, update);  
  
 System.*out*.println("------------------One monitor update-------------------");  
 System.*out*.println(Collection.find(filter).first().toJson(prettyPrint));  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void deleteMonitor(Monitor monitor) {  
 try (MongoClient mongoClient = MongoClients.*create*("mongodb://127.0.0.1:27017")) {  
 MongoDatabase DB = mongoClient.getDatabase("Monitors");  
 MongoCollection<Document> Collection = DB.getCollection("Monitor");  
  
 Bson filter = *eq*("\_id", monitor.getId());  
 DeleteResult result = Collection.deleteOne(filter);  
 System.*out*.println(result);  
 System.*out*.println("------------------One monitor delete-------------------");  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

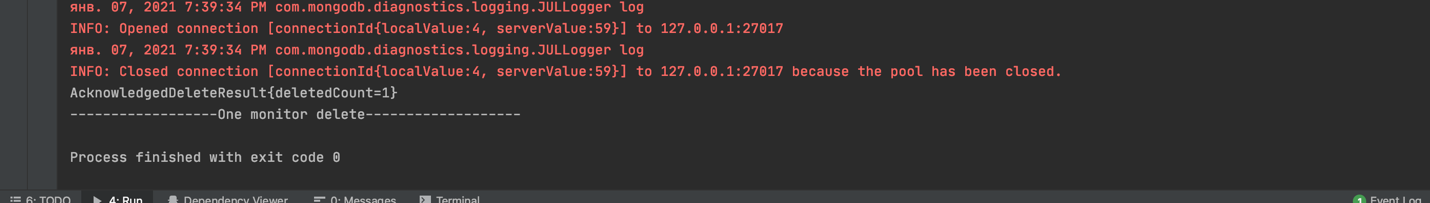
**Monitor**

package Models;  
  
  
import org.bson.Document;  
import org.bson.types.ObjectId;  
  
import java.util.List;  
import java.util.Objects;  
  
public class Monitor {  
  
 private ObjectId id;  
 private String Name;  
 private String Matrix\_type;  
 private String Aspect\_ratio;  
 private int Diagonal\_matrix;  
 private int Screen\_resolution;  
 private int Screen\_refresh\_rate;  
 public Type Type;  
 public Model Model;  
  
 public Monitor(ObjectId id, String name, String matrix\_type,  
 String aspect\_ratio, int diagonal\_matrix, int screen\_resolution,  
 int screen\_refresh\_rate, Type type, Model model) {  
 this.id = id;  
 Name = name;  
 Matrix\_type = matrix\_type;  
 Aspect\_ratio = aspect\_ratio;  
 Diagonal\_matrix = diagonal\_matrix;  
 Screen\_resolution = screen\_resolution;  
 Screen\_refresh\_rate = screen\_refresh\_rate;  
 Type = type;  
 Model = model;  
 }  
  
 public Monitor(String name, String matrix\_type, String aspect\_ratio,  
 int diagonal\_matrix, int screen\_resolution, int screen\_refresh\_rate,  
 Type type, Model model) {  
 Name = name;  
 Matrix\_type = matrix\_type;  
 Aspect\_ratio = aspect\_ratio;  
 Diagonal\_matrix = diagonal\_matrix;  
 Screen\_resolution = screen\_resolution;  
 Screen\_refresh\_rate = screen\_refresh\_rate;  
 Type = type;  
 Model = model;  
 }  
  
 public Monitor(ObjectId id) {  
 this.id = id;  
 }  
  
 public ObjectId getId() {  
 return id;  
 }  
  
 public Monitor setId(ObjectId id) {  
 this.id = id;  
 return this;  
 }  
  
 public String getName() {  
 return Name;  
 }  
  
 public Monitor setName(String name) {  
 this.Name = name;  
 return this;  
 }  
  
 public String getMatrix\_type() {  
 return Matrix\_type;  
 }  
  
 public Monitor setMatrix\_type(String matrix\_type) {  
 this.Matrix\_type = matrix\_type;  
 return this;  
 }  
  
 public String getAspect\_ratio() {  
 return Aspect\_ratio;  
 }  
  
 public Monitor setAspect\_ratio(String aspect\_ratio) {  
 this.Aspect\_ratio = aspect\_ratio;  
 return this;  
 }  
  
 public int getDiagonal\_matrix() {  
 return Diagonal\_matrix;  
 }  
  
 public Monitor setDiagonal\_matrix(int diagonal\_matrix) {  
 this.Diagonal\_matrix = diagonal\_matrix;  
 return this;  
 }  
  
 public int getScreen\_resolution() {  
 return Screen\_resolution;  
 }  
  
 public Monitor setScreen\_resolution(int screen\_resolution) {  
 this.Screen\_resolution = screen\_resolution;  
 return this;  
 }  
  
 public int getScreen\_refresh\_rate() {  
 return Screen\_refresh\_rate;  
 }  
  
 public Monitor setScreen\_refresh\_rate(int screen\_refresh\_rate) {  
 this.Screen\_refresh\_rate = screen\_refresh\_rate;  
 return this;  
 }  
  
 public Type getType() {  
 return Type;  
 }  
  
 public void setType(Type type) {  
 Type = type;  
 }  
  
 public Model getModel() {  
 return Model;  
 }  
  
 public void setModel(Model model) {  
 Model = model;  
 }  
  
 @Override  
 public String toString() {  
 final StringBuffer sb = new StringBuffer("Monitor{");  
 sb.append("id=").append(id);  
 sb.append(", Name='").append(Name).append('\'');  
 sb.append(", Matrix\_type='").append(Matrix\_type).append('\'');  
 sb.append(", Aspect\_ratio='").append(Aspect\_ratio).append('\'');  
 sb.append(", Diagonal\_matrix=").append(Diagonal\_matrix);  
 sb.append(", Screen\_resolution=").append(Screen\_resolution);  
 sb.append(", Screen\_refresh\_rate=").append(Screen\_refresh\_rate);  
 sb.append(", Type=").append(Type);  
 sb.append(", Model=").append(Model);  
 sb.append('}');  
 return sb.toString();  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Monitor monitor = (Monitor) o;  
 return Diagonal\_matrix == monitor.Diagonal\_matrix &&  
 Screen\_resolution == monitor.Screen\_resolution &&  
 Screen\_refresh\_rate == monitor.Screen\_refresh\_rate &&  
 id.equals(monitor.id) &&  
 Name.equals(monitor.Name) &&  
 Matrix\_type.equals(monitor.Matrix\_type) &&  
 Aspect\_ratio.equals(monitor.Aspect\_ratio) &&  
 Type.equals(monitor.Type) &&  
 Model.equals(monitor.Model);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(id, Name, Matrix\_type, Aspect\_ratio, Diagonal\_matrix, Screen\_resolution, Screen\_refresh\_rate, Type, Model);  
 }  
}

**Main**

import Dao.ModelDao;  
import Dao.MonitorDao;  
import Dao.TypeDao;  
import Impl.ModelDaoImpl;  
import Impl.MonitorDaoImpl;  
import Impl.TypeDaoImpl;  
import Models.Model;  
import Models.Monitor;  
import Models.Type;  
import com.mongodb.client.MongoClient;  
import com.mongodb.client.MongoClients;  
import org.bson.Document;  
import org.bson.json.JsonWriterSettings;  
import org.bson.types.ObjectId;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import static java.util.Arrays.*asList*;  
  
public class Main {  
 public static void main(String[] args) {  
  
 //insert model  
// Model model = new Model("Test","Test", "Test");  
// ModelDao modelDao = new ModelDaoImpl();  
// modelDao.createModel(model);  
  
 //insert type  
// Type type = new Type("Test", "Test");  
// TypeDao typeDao = new TypeDaoImpl();  
// typeDao.createType(type);  
  
 //insert monitor  
// Model model = new Model("Test","Test", "Test");  
// Type type = new Type("Test", "Test");  
// Monitor monitor = new Monitor("Test", "Test", "Test", 21, 1080, 60, type, model);  
// MonitorDao monitorDao = new MonitorDaoImpl();  
// monitorDao.createMonitor(monitor);  
  
 //find by name model  
// ModelDao modelDao = new ModelDaoImpl();  
// modelDao.getByModel("Test");  
  
 //find by type  
// TypeDao typeDao = new TypeDaoImpl();  
// typeDao.getByNameType("Custom monitor");  
  
 //find by name monitor  
// MonitorDao monitorDao = new MonitorDaoImpl();  
// monitorDao.getByName("Taya");  
  
 //find all models  
// ModelDao modelDao = new ModelDaoImpl();  
// modelDao.getAllModels();  
  
 //find all types  
// TypeDao typeDao = new TypeDaoImpl();  
// typeDao.getAllTypes();  
  
 //find all monitors  
// MonitorDao monitorDao = new MonitorDaoImpl();  
// monitorDao.getAllMonitors();  
  
 //update model  
// ObjectId id = new ObjectId("5ff1f52b7f11f303c4b7ceaa");  
// Model model = new Model(id,"Test1", "Test1", "Tes1");  
// ModelDao modelDao = new ModelDaoImpl();  
// modelDao.updateModel(model);  
  
 //update type  
// ObjectId id = new ObjectId("5ff71eb0ed8d8b767e7e27f5");  
// Type type = new Type(id,"Test1","Test1");  
// TypeDao typeDao = new TypeDaoImpl();  
// typeDao.updateType(type);  
  
 //update monitor  
 ObjectId id = new ObjectId("5ff746f8dbce6a593e219758");  
 Model model1 = new Model("Test","Test", "Test");  
 Type type1 = new Type("Test", "Test");  
 Monitor monitor1 = new Monitor(id,"Test1", "Test1", "Test1", 21, 1080, 60, type1, model1);  
 MonitorDao monitorDao1 = new MonitorDaoImpl();  
 monitorDao1.updateMonitor(monitor1);  
  
 //delete model  
// ObjectId id = new ObjectId("5ff1f52b7f11f303c4b7ceaa");  
// Model model = new Model(id);  
// ModelDao modelDao = new ModelDaoImpl();  
// modelDao.deleteModel(model);  
  
 //delete type  
// ObjectId id = new ObjectId("5ff71eb0ed8d8b767e7e27f5");  
// Type type = new Type(id);  
// TypeDao typeDao = new TypeDaoImpl();  
// typeDao.deleteType(type);  
  
 //delete monitor  
 ObjectId id2 = new ObjectId("5ff746f8dbce6a593e219758");  
 Monitor monitor2 = new Monitor(id2);  
 MonitorDao monitorDao2 = new MonitorDaoImpl();  
 monitorDao2.deleteMonitor(monitor2);  
 }  
}

****

****

**5.Висновки**

В ході виконання лабораторних роботи була спроектована і розроблена база даних з відповідною структурою до обраної предметної області. Отримали практичні навички: аналізу предметної області та проектування на цій основі структури БД.